UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PPGSESAI11 - LABORATÓRIO DE REDE DE SENSORES SEM FIO
ATIVIDADE DE LABORATÓRIO

Professor: Guilherme Luiz Moritz e Ohara Kerusauskas Rayel
Tema: Simulação de redes utilizando Cooja
Data: Setembro de 2016

Introdução

Existem diversas situações onde uma simulação computacional pode ser útil no desenvolvimento de uma rede de sensores sem fio:

- Redes complexas podem se tornar difíceis de serem depuradas;
- Os problemas observados em redes reais podem ser difíceis de serem reproduzidos;
- Algoritmos complexos de roteamento eu economia de energia podem ser difíceis de serem avaliados;

O Cooja é uma plataforma que é parte integrante do Contiki e pode ser utilizada para simulação de redes. A grande vantagem desta plataforma é que ela executa o mesmo *firmware* que seria gravado num nó de *hardware* real. Infelizmente não há suporte para simulações de *cores* 8051 porém é possível desenvolver código que execute tanto em um core MSP quanto no core alvo da disciplina.

Atividade 1 - Simulação simples UDP Client/Server

Nesta atividade, utiliza-se o Cooja para executar-se um exemplo simples de transmissão UDP. A atividade é baseada em [1].

1. Alterne para o diretório do Cooja (contiki/tools/cooja);
2. Execute o Cooja com o comando 'ant run'; Deve ser observada a interface inicial do Cooja, conforme Figura 1;

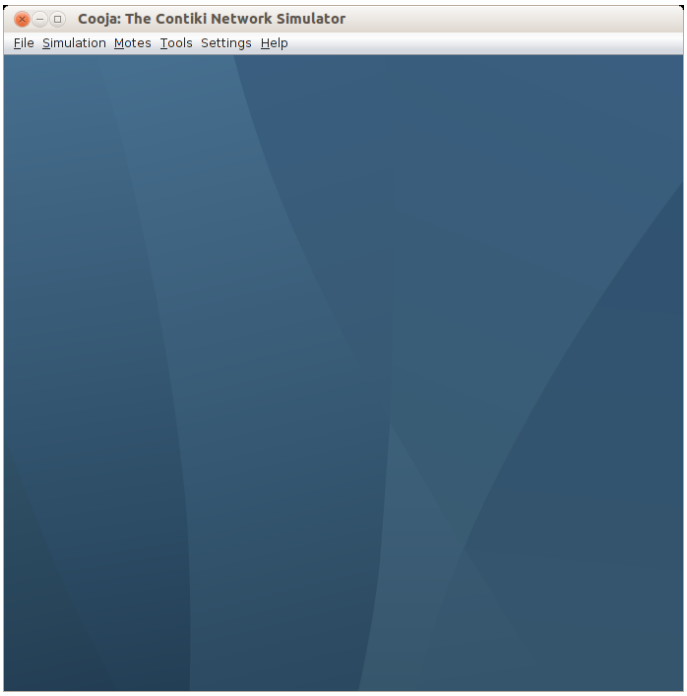

Figura 1: Tela inicial do Cooja

3. Crie uma nova simulação através do menu File→New Simulation;
4. Configure os parâmetros de simulação conforme a Figura 2;

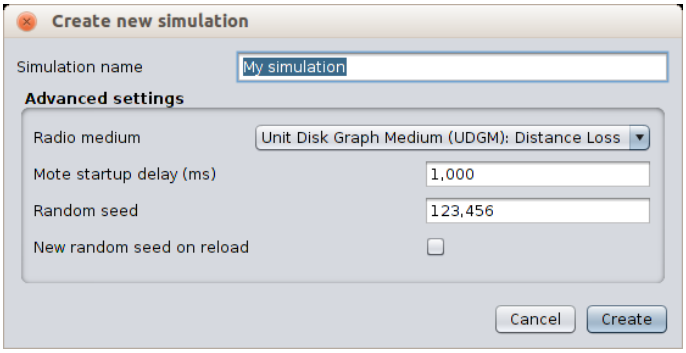

Figura 2: Nova simulação no Cooja

5. Adicione um novo nó à simulação. Utilize o menu Motes→Create New Mote Type→Sky mote... Observe que infelizmente não há disponível um nó do tipo CC2530 e, por este motivo, a simulação será realizada com um Tmote Sky;

6. Para a aplicação UDP, dois tipos de nós serão adicionados. O primeiro modelo executará o *firmware* de um servidor UDP. Utilize o botão browse para encontrar o arquivo examples/ipv6/rpl-udp/udp-server.c; Nomeie o nó como receiver

7. Pressione o botão 'Compile' para compilar o exemplo para a plataforma adequada;

8. Após encerrada a compilação, pressione o botão 'Create';

9. Adicione 1 nó do tipo;

10. Na janela Network, utilize o menu View para configurar a seguintes propriedades do nó:

    (a) Mote IDs;
    (b) Addresses: IP or Rime;
    (c) Radio Traffic;
    (d) 10m background grid;
    (e) Radio Environment.

11. Adicione um novo tipo de nó sky. Nomeie o nó como sender; Utilize como *firmware* o arquivo examples/ipv6/rpl-udp/udp-client.c;

12. Execute a simulação. Observe que é possível observar a saída dos terminais dos nós na janela Mote Output.

13. Também é possível controlar a velocidade de simulação na janela Simulation Control. Altere a opção Speed Limit para 100%.

14. Clique na representação do nó. Observe que é exibido um círculo verde que indica o alcance do radio. Mova os nós para que os mesmos fiquem fora do alcance e observe as mensagens no terminal;

15. Adicione um novo nó do tipo receiver, mova o para uma posição intermediária entre os primeiros nós adicionados. Observe as mensagens no terminal e verifique se a comunicação foi restabelecida com a formação de uma rede mesh.

Atividade 2 - Border Router Simulado

O Border Router é um nó que é instalado na ponta de uma rede de sensores para que a mesma tenha acesso a outras redes. Este nó é peça fundamental para que uma rede de sensores possa acessar a internet. Uma vez que se crie uma ponte IP entre a rede de sensores e um dispositivo IP com acesso à internet, todos os nós ficam acessíveis e tem acesso à rede exterior. A representação de uma rede de sensores com Border Router pode ser observada na Figura 3.

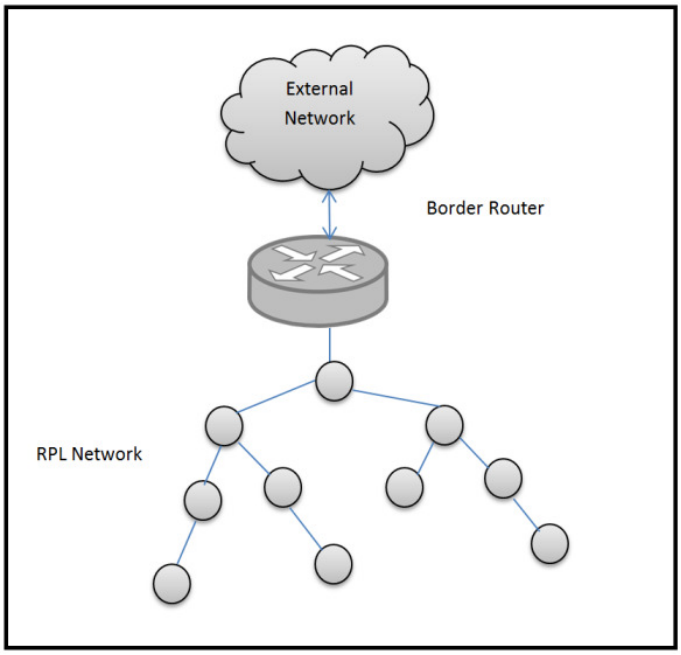

Figura 3: Border Router [2]

Na implementação que será utilizada no Contiki, o border router é responsável por criar a raiz de roteamento (DAG root), e, consequentemente, todos os pacotes que não são direcionados a outros nós da rede de sensores serão roteados para o border router. Um pacote recebido pelo DAG root mas que não é direcionado à rede de sensores criada é enviado para uma Porta Serial Sobre USB que está conectada a um PC executando Linux. Um software denominado tunslip6 é responsável por receber os pacotes IP via Porta Serial e encaminhá-los para uma interface TUN [3].

Assim que o pacote IP for encaminhado à interface TUN ele é tratado como qualquer pacote IP que trafegue pelo Kernel Linux. Todas as regras avançadas de roteamento e *firewall* se aplicam (o assunto de configuração de redes e roteamento Linux está fora do escopo da disciplina).

No primeiro experimento será criada uma rede de sensores sem fio simulada com o auxílio do Cooja. Um dos nós desta rede estará executando um *firmware* de border router. O Linux executará o software tunslip6 e abrirá uma porta serial emulada pelo Cooja. Desta maneira será possível acessar os nós sensores através do Linux, e consequentemente de qualquer dispositivo que possuir uma rota válida para o Kernel Linux que executa na máquina virtual (este tutorial é baseado em [2]).

1. Incremente a simulação da primeira atividade, criando um nó do tipo border router, cujo código se encontra em examples/ipv6/rpl-border-router/border-router.c;

2. Compile o tunslip6 conforme instruções disponíveis em [4];

3. Observe que o tunslip6 é programado para encaminhar pacotes entre porta serial e uma interface TUN. Como neste exemplo será utilizada uma rede simulada, é necessário que se crie uma porta serial virtual (que executará sobre TCP). Clique com o botão oposto nó que está executando o *firmware* do border router e selecione a opção Mote tools→Serial Socket (SERVER). Configurar o socket de acordo com a Figura 4;

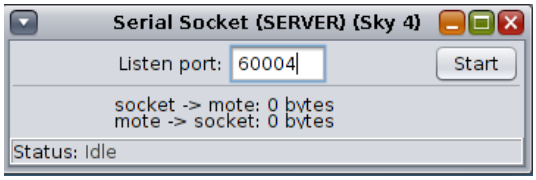

Figura 4: Socket serial do Cooja

4. Reinicie a simulação utilizando a opção Reload e depois Start da janela Simulation Control;
5. Conecte o tunslip6 à rede simulada utilizando o comando sudo ./tunslip6 -a 127.0.0.1 aaaa::1/64
6. Na saída do terminal do tunslip6, observe as configurações de ip da interface tun0, além dos IPs do border router. Porque existem dois endereços IP? Quais são suas diferenças?
7. Numa nova janela do terminal, utilize o comando ping6 para testar a conectividade do Kernel Linux com a rede criada (envie um pacote de ping para o border router). Se o ping for bem sucedido, qualquer tráfego IP pode ser enviado ao nó (observar que o nó não suporta TCP).
8. Envie um pacote de ping para o nó sender (utilize o menu View→Addresses: IP or Rime para obter o IP link local dos nós); O comando obteve sucesso? Porquê? Observe os tempos de retorno do comando ping. Qual o motivo deles serem diferentes dos retornos de ping para o border router?
9. Envie um pacote de ping para o nó receiver; O comando obteve sucesso? Porquê?
10. Acesse o IP do border router utilizando o browser do Linux. O que é possível observar? (o endereço deve ser digitado entre colchetes)
11. Utilizando o Windows hospedeiro, envie um pacote de ping para o border router. O comando foi bem sucedido? Repita o envio porém agora enviando um pacote de ping para o Linux Hospedado. O comando foi bem sucedido? Porquê?

Atividade 3 - Border Router

Neste experimento o nó da disciplina será conectado numa rede de sensores sem fio com border router. Como cada equipe só possui um nó, somente uma rede será formada, sendo o border router responsabilidade do professor.

Um nó conectado ao border router é capaz de acessar qualquer endereço IP que seja permitido pelas regras de roteamento do Linux. No caso do experimento, todos os computadores do laboratório estarão conectados em uma rede local IPv6 (alguma com prefixo bbbb::/64). O Linux conectado ao border router anunciará uma rota para a rede de sensores configurada e desta maneira todos os dispositivos IPv6 da sala estarão acessíveis através da rede.

Neste momento, a conectividade da rede será verificada com a criação de um script em Linux que enviará um pacote UDP para o nó conectado, alterando o estado do led.

Num segundo momento, o *firmware* do kit será alterado para enviar um pacote UDP a ser recebido por um script no Linux.

1. Para todos os envios e recebimentos, utilize a porta 8802;
2. Defina as seguintes macros:

```
#define LED_TOGGLE_REQUEST (0x79)
#define LED_SET_STATE (0x7A)
#define LED_GET_STATE (0x7B)
#define LED_STATE (0x7C)
```

3. Imprima no terminal o IPv6 do nó programado;
4. Envie um comando de ping para o IP do Linux que está executando o software do border router; Observe o tempo de retorno;
5. Verifique a tabela de roteamento com o comando ip -6 route show
6. Envie um comando de ping para o IP do border router; Observe o tempo de retorno e o TTL; O comando foi bem sucedido? Porque?
7. Envie um comando de ping para o IP do Gateway (bbbb::100); Observe o tempo de retorno e o TTL; O comando foi bem sucedido? Porque?
8. Adicione uma rota para a sub-rede aaaa:: para o Gateway da Rede de Sensores com o comando sudo route -A inet6 add aaaa::/64 gw bbbb::100
9. Envie um comando de ping para o IP do border router; Observe o tempo de retorno e o TTL; O comando foi bem sucedido? Porque?
10. Envie um comando de ping para o IP do nó do seu kit; Observe o tempo de retorno e o TTL;
11. Escreva um script que envie pacote UDP para alterar o estado dos leds do kit após receber um pacote UDP de 1 byte contendo a macro `LED_TOGGLE_REQUEST`. Recomenda-se o uso de Python. Um exemplo de envio de pacote UDP segue:

```
import socket
UDP_PORT = 3000
UDP_IP = "aaaa::212:4b00:7c3:b5bb"
MESSAGE = "Sending␣an␣UDP␣packet␣using␣python!"
print "UDP␣target␣IP:", UDP_IP
print "UDP␣target␣port:", UDP_PORT
print "message:", MESSAGE
# socket.SOCK_DGRAM = UDP connection
sock = socket.socket(socket.AF_INET6,socket.SOCK_DGRAM)
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

O pacote de alteração do estado do led pode é definido como:

Valor	LED_SET_STATE	Macro de indicação de led	Estado
byte	0	1	2

12. Escreva um script que escute mensagens recebidas na porta estipulada, e imprima no terminal o estado dos leds do kit assim que receba um pacote de `LED_STATE`.

```
import socket
import struct
import random

HOST = '' #all interfaces
UDP_PORT = 8802

sock = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM) # UDP
sock.bind((HOST, UDP_PORT))

LED_TOGGLE_REQUEST = (0x79)
LED_SET_STATE = (0x7A)
LED_GET_STATE = (0x7B)
LED_STATE = (0x7C)

while True:
        data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
        print "received message:", data, "from: [", addr[0].strip(),"]:",addr[1]
                offset = 0
        op = struct.unpack_from(">B", data,offset)
        offset += struct.calcsize(">B")

        if op[0] == LED_TOGGLE_REQUEST:
            print "RECEIVED LED_TOGGLE_REQUEST , SEND LED_TOGGLE TO [", addr[0].strip(),"]:",addr[1]
            stt = random.randint(0, 15)
            msg = struct.pack(">BB", LED_SET_STATE, stt)

        if op[0] == LED_STATE:
            state = struct.unpack_from(">B", data, offset)
```

O pacote de led state é definido como:

Valor	LED_STATE	Macro de indicação de estado dos leds
byte	0	1

13. Altere o firmware da Atividade 4 para ingressar na PAN do novo border router. Defina a macro `#define IEEE802154_CONF_PANID` para `0xABCD` (/platform/cc2530dk/contiki-conf.h)

14. Altere o firmware da Atividade 4 para que os pacotes sejam enviados para o PC que executa o script desenvolvido nos itens 11 e 12.

15. Verifique o funcionamento da solução.

Referências

[1] Contiki OS, "Get started with Contiki, Instant Contiki and Cooja," http://www.contiki-os.org/start.html, Março 2015, acessado em 2016-08-18.

[2] Contiki OS, "RPL Border Router - contiki," http://anrg.usc.edu/contiki/index.php/RPL_Border_Router, Março 2015, acessado em 2016-08-04.

[3] D. Simic, "TUN/TAP - Wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/TUN/TAP, Julho 2016, acessado em 2016-08-04.

[4] FIT IoT lab, "Build tunslip6 - fit/iot-lab," https://www.iot-lab.info/tutorials/build-tunslip6/, Abril 2012, acessado em 2016-08-04.